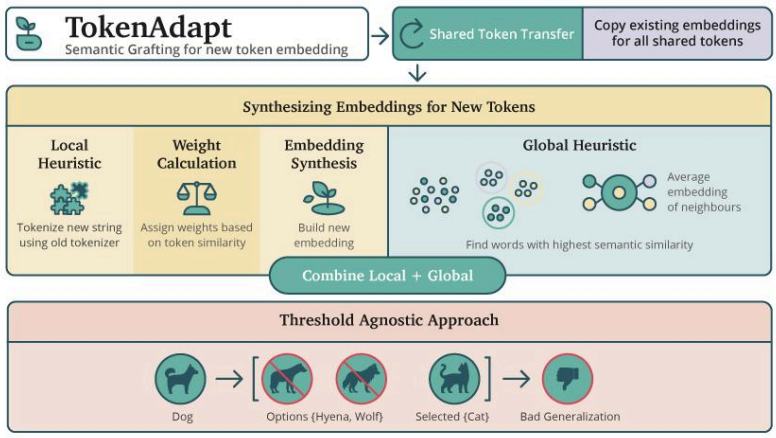
> Here's the **pre-print** of our work on tokenadapt and supertokenizer: https://arxiv.org/abs/2505.09738v1

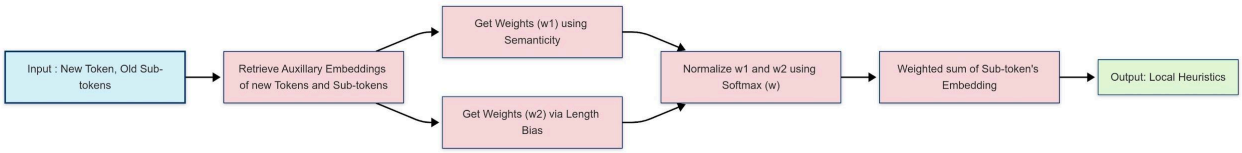
> llms face constraints due to fixed tokenization schemes, especially in specialized domains. we created a framework for efficient tokenizer transplantation with with high fidelity semantic preservation & minimal retraining.

> suboptimal tokenization causes fragmentation, inflating computational costs & degrading semantic coherence. our framework directly addresses this by enabling seamless adoption.

> our method employs a novel hybrid(local + global) heuristic to initialize embeddings for new, unique tokens, ensuring semantic continuity and preserving the base model's learned representations when adopting a new tokenizer.



> local heuristic: new tokens are split by the original tokenizer. an external embedding model compares semantic similarity between the full token and its sub tokens. these similarities weight the original sub token embeddings, with length normalization boosting effectiveness.


Input : New Token, Old Sub-tokens
Retrieve Auxillary Embeddings of new Tokens and Sub-tokens
Get Weights (w1) using Semanticity
Get Weights (w2) via Length Bias
Normalize w1 and w2 using Softmax (w)
Weighted sum of Sub-token's Embedding
Output: Local Heuristics

> global heuristic: using the same external embedding space and efficient vector search, we identify the nearest neighbors for the new token within the entire original vocab based on semantic similarity. original embeddings of these neighbours are averaged, weighted by similarity.

Input : New Token, Old Vocab
Normalize Embedding for New Token
Search K Nearest Neighbours in Old Vocab
Weight K Tokens using Similarities and Softmax
Weighted Sum of K Token's Embedding
Output: Global Heuristics

> our investigations validate that the transplantation heuristic successfully initializes unique tokens, outperforming conventional baselines and sophisticated methods including transtokenizer and retok.

Table 1: Zero-Shot Perplexity Ratio Comparison - Lower is Better

Experiment Context	Initialization Method	Overall Mean Ratio	English Mean Ratio	Code Mean Ratio	Math Mean Ratio	Hinglish Mean Ratio	Hindi Mean Ratio
Base: Llama-3.2-3B → Target: QTK-81K							
	TokenAdapt (Local Only)	64.4	**1.21**	1.40	2.02	14.3	1682.4
	TokenAdapt (Hybrid)	**48.2**	1.29	1.35	2.16	**13.6**	1108.4
	TokenAdapt (Hybrid + Thr.)	**48.2**	1.28	**1.34**	**1.53**	15.97	1105.1
	ReTok Baseline	71.1	1.29	1.42	1.54	14.7	1875.6
	TransTokenizer Baseline	145.9	94.1	15.4	73.8	125.1	**1052.2**
Base: Qwen2.5-3B → Target: QTK-81K							
	TokenAdapt (Local Only)	109.3	1.44	2.27	1.64	**12.6**	6149.4
	TokenAdapt (Hybrid)	85.8	1.39	**1.86**	**1.37**	13.4	4615.9
	TokenAdapt (Hybrid + Thr.)	**85.04**	**1.38**	1.91	2.00	13.44	**775.42**
	ReTok Baseline	139.1	1.43	2.28	1.64	12.8	8007.9
	TransTokenizer	101.6	103.0	23.0	100.4	83.0	1302.1
Base: Llama-3.2-3B → Target: Adi-Bun-128K (ST)							
	TokenAdapt (Local Only)	1371.3	733.1	87.2	**71.0**	236.2	41.46
	TokenAdapt (Hybrid)	**577.5**	376.1	93.6	78.2	**105.6**	**13.12**
	TokenAdapt (Hybrid + Thr.)	589.03	**256.23**	**46.06**	194.98	550.37	6997.74
	ReTok Baseline	1174.0	1749.8	134.0	806.6	9249.5	9279.7
	TransTokenizer Baseline	4173.6	22005.2	1284.6	6110.5	32117.8	10818.8

PPL Ratio = (Transplanted Mean PPL / Original Base Model Mean PPL). Lower ratios indicate less degradation. See Sec. 4 for base PPL values and specific configurations. Parallel data used for Transtokenization experiments were: allenai/nllb and open_subtitles. Bolder datapoint is best result. Underlined is the 2nd best result.

> the paper also introduce supertokenizer. our goal is to create single tokens representing multiple words. unlike some methods needing whole words for multi word tokens, our approach is flexible. a super token combine words and subwords ("frequently_us" from "frequently used").

Table 3: Comparison of text segmentation by different tokenization strategies.

Tokenizer	Tokenization
GPT4o (203 Bytes, 43 Tokens: 4.721 Bytes/token)	The problem asks for the number of rational numbers between 0 and 1 such that when the rational number is written as a fraction in lowest terms, the product of the numerator and the denominator is $20!$.
Llama3 (203 Bytes, 43 Tokens: 4.721 Bytes/token)	The problem asks for the number of rational numbers between 0 and 1 such that when the rational number is written as a fraction in lowest terms, the product of the numerator and the denominator is $20!$.
SuperBPE 200K (203 Bytes, 30 Tokens: 6.767 Bytes/token)	The problem asks for the number of rational numbers between 0 and 1 such that when the rational number is written as a fraction in lowest terms, the product of the numerator and the denominator is $20!$.
Adibun 128K (Ours) (203 Bytes, 32 Tokens: 6.344 Bytes/token)	The problem asks for the number of rational numbers between 0 and 1 such that when the rational number is written as a fraction in lowest terms, the product of the numerator and the denominator is $20!$.

> the main concept is using randomization guided by a probability distribution for chunk length. we augment the dataset to create random chunking using this distribution, not based on whitespace splitting.

> the random chunking allows common multi word sequences (or word + subword sequences) to the bpe trainer and it automatically merges them based on frequency.

> supertokens aim to capture more meaningful semantic units. empirical results demonstrate notable compression gains across diverse domains with our supertoken trained tokenizer.

Table 2: Tokenizer Compression Comparison Across Domains (Total Tokens Used - Lower is Better)

Tokenizer	English	Hindi	Python	Java	C++	Math
Adi-Bun (128k)[a]	**283,657**	**9,174**	**4,765**	**1,821**	**11,464**	**3,887**
DeepSeek-R1 (128k)[b]	338,873	22,854	5,270	2,223	13,081	5,375
Krutrim-Ins (131k)[c]	343,067	15,477	5,246	2,223	13,007	5,361
Gemma-3-27b(262k)[d]	329,308	10,833	5,998	2,590	16,175	5,867

[a] `tinycompany/Adi-Bun-128K` [b] `deepseek-ai/DeepSeek-R1` (128k variant)
[c] `krutrim-ai-labs/Krutrim-2-Instruct`
[d] `unsloth/Gemma-3-27b-it-unsloth-bnb-4bit`

> our method efficiently overcomes tokenizer limits and it helps llms adapt better to specialized tasks and low resource settings. this marks our first attempt in presenting our research this way and we're committed to improving with future work.